**Clustering Schoolwork**

**任务一：将鸢尾花数据集画成图的形式。**

1.借助sklearn包，导入数据集

2.利用逻辑回归模型训练样本集

3.绘制散点图

**任务二：确定一个合适的阈值，只有两个样本之间的相似度大于该阈值时，这两个样本之间才有一条边。**

**任务三：求取带权邻接矩阵。**

###1.计算距离矩阵

def euclidDistance(x1,x2,sqrt\_flag=False):

res = np.sum((x1-x2)\*\*2) 这么计算距离正好对应求高斯核的公式

if sqrt\_flag:

res = np.sqrt(res)

return res

def calEuclidDistanceMatrix(X):

X = np.array(X)

S = np.zeros((len(X), len(X)))

for i in range(len(X)):

for j in range(i+1, len(X)):

S[i][j] = 1.0 \* euclidDistance(X[i], X[j])

S[j][i] = S[i][j] 距离矩阵是对称阵

return S

###2.利用knn计算邻接矩阵

根据距离矩阵，计算每个点的前k个近邻，在对这个点与k个近邻的距离带入高斯核的公式，得到带权邻接矩阵。

带权邻接矩阵中为0的代表两个点之间没有边，相反有边

sigma在0.1到10之间试的

def myKNN(S,k,sigma=0.187):

N=len(S)

A=np.zeros((N,N))

for i in range(N):

dist\_with\_index=zip(S[i],range(N))#带索引的距离

dist\_with\_index=sorted(dist\_with\_index,key=lambda x:x[0])#按距离从小到大排序

neighbours\_id=[dist\_with\_index[m][1] for m in range(k+1)]#得到前k个最近邻的索引

# xi's k nearest neighbours

for j in neighbours\_id:# xj is xi's neighbour

A[i][j] = np.exp(-S[i][j]/2/sigma/sigma)#高斯核

A[j][i] = A[i][j] # mutually 邻接矩阵是对称的

return A

**任务四：根据邻接矩阵进行聚类。**

谱聚类实现过程

11.计算距离矩阵

22.利用KNN计算邻接矩阵A

33.由A计算度数矩阵D和拉普拉斯矩阵L

44.标准化L=D^(-1/2) L D^(-1/2)

55.对称阵L=D^(-1/2) L D^(-1/2)进行特征值分解，得到特征向量Hnn

66.将Hnn当成样本送入Kmeans

77.获得聚类结果C=（C1，C2，....Ck）

###3标准化的拉普拉斯矩阵(以上两步，前两个任务就完成了)

def calLaplacianMatrix(adjacentMatrix):

#compute the Degree Matrix:D=sum(A)

degreeMatrix=np.sum(adjacentMatrix,axis=1)

#compute the Laplacian Matrix:L=D-A

laplacianMatrix=np.diag(degreeMatrix)-adjacentMatrix#度数矩阵是对角矩阵

#normailize

# D^(-1/2) L D^(-1/2)

sqrtDegreeMatrix = np.diag(1.0 / (degreeMatrix \*\* (0.5)))

return np.dot(np.dot(sqrtDegreeMatrix, laplacianMatrix), sqrtDegreeMatrix)

###5.对称阵L=D^(-1/2) L D^(-1/2)进行特征值分解，得到特征向量Hnn

###6.将Hnn当成样本送入Kmeans

eig\_x,V= np.linalg.eig(Laplacian) # H'shape is n*n# H'shape is n*n

eig\_x=np.array(eig\_x,dtype=np.float32)

V=np.array(V,dtype=np.float32)

eig\_x=zip(eig\_x,range(len(x)))

eig\_x=sorted(eig\_x, key=lambda eig\_x:eig\_x[0])特征值从大到小排列

H = np.vstack([V[:,i] for (v, i) in eig\_x[:6]]).T#(shape (150,6))

#试了试是前6个正确率最高

#将H按行进行归一化

Hmax,Hmin=H.max(axis=0),H.min(axis=0)

H=(H-Hmin)/(Hmax-Hmin)

7.获得聚类结果C=（C1，C2，....Ck）

sp\_kmeans=KMeans(n\_clusters=3,n\_init=13,random\_state=17).fit(H)

**任务五：将聚类结果可视化，重新转换成图的形式，其中每一个簇应该用一种形状表示，比如分别用圆圈、三角和矩阵表示各个簇**

N=nx.Graph()

for i in range(150):

N.add\_node(i)

#print(N.nodes())

edglist=[]

根据邻接矩阵，a[i][j]>0说明两个点之间有边

for i in range(150):

for j in range(150):

if(A[i][j]>0):

edglist.append((i,j))

#print(edglist)

三类对应三种不同的颜色

colorlist=[]

N.add\_edges\_from(edglist)

for i in range(150):

if(y[i]==0):

colorlist.append('r')

elif(y[i]==1):

colorlist.append('b')

else:

colorlist.append('orange')

#print(N.neighbors((data\_new[1][0],data\_new[1][1])))

#print(N[(data\_new[1][0],data\_new[1][1])])

nx.draw(N,pos = nx.circular\_layout(N),node\_color = colorlist,edge\_color = 'black',with\_labels = False,font\_size =5,node\_shape='o' ,node\_size

=25,width=0.3)

plt.show()

**任务六：求得分簇正确率**

#计算正确率

count=0

for i in range(150):

if y[i]==sp\_kmeans.labels\_[i]:

count=count+1

print('Accuracy:',100\*(count/150),'%')

目前得到的正确率为81.3333333%

**任务七：完成代码的描述文档**

已完成